PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: The ACM Digital Library The Guide

await advance instruction <and> CFG <and> basic block" <an SEARCH

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfaction survey

Terms used: **await advance instruction** and **CFG** and **basic block** and **instruction hoisting**

Found **136,978** of **206,720**

Sort results by: relevance

Display results: expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

Relevance scale

1 Automatic parallelization: Automatic multithreading and multiprocessing of C programs for IXP

Long Li, Bo Huang, Jinquan Dai, Luddy Harrison

June 2005 **Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming PPoPP '05**

Publisher: ACM Press

Full text available: pdf(634.36 KB) Additional Information: full citation, abstract, references, index terms

Effective compilation of packet processing applications onto the Intel IXP network processors requires, among other things, the automatic use of multiple threads on one or more processing elements, and the automatic introduction of synchronization as required to correctly enforce dependences between such threads. We describe the program transformation that is used in the Intel Auto-partitioning C Compiler for IXP to automatically multithread/multi-process a program for the IXP. This transformati ...

Keywords: code motion, critical section, multi-processing, multi-threading, network processor

2 Instruction fetch and control flow: Reducing control overhead in dataflow architectures

Andrew Petersen, Andrew Putnam, Martha Mercaldi, Andrew Schwerin, Susan Eggers, Steve Swanson, Mark Oskin

September 2006 **Proceedings of the 15th international conference on Parallel architectures and compilation techniques PACT '06**

Publisher: ACM Press

Full text available: pdf(662.72 KB) Additional Information: full citation, abstract, references, index terms

In recent years, computer architects have proposed tiled architectures in response to several emerging problems in processor design, such as design complexity, wire delay, and fabrication reliability. One of these architectures, WaveScalar, uses a dynamic, tagged-token dataflow execution model to simplify the design of the processor tiles and their interconnection network and to achieve good parallel performance. However, using a dataflow execution model reawakens old problems, including the ins ...

Keywords: Wavescalar, compiler, dataflow, tiled architecture

3 The theory of parsing, translation, and compiling

Alfred V. Aho, Jeffrey D. Ullman
January 1972 Book

Publisher: Prentice-Hall, Inc.

Full text available: pdf(98.28 MB) Additional Information: full citation, abstract, references, cited by, index terms

From volume 1 Preface (See Front Matter for full Preface)

This book is intended for a one or two semester course in compiling theory at the senior or graduate level. It is a theoretically oriented treatment of a practical subject. Our motivation for making it so is threefold.

(1) In an area as rapidly changing as Computer Science, sound pedagogy demands that courses emphasize ideas, rather than implementation details. It is our hope that the algorithms and concepts presen ...

4 Multiscalar processors

Gurindar S. Sohi, Scott E. Breach, T. N. Vijaykumar
May 1995 **ACM SIGARCH Computer Architecture News , Proceedings of the 22nd annual international symposium on Computer architecture ISCA '95**, Volume 23 Issue 2

Publisher: ACM Press

Full text available: pdf(1.44 MB) Additional Information: full citation, abstract, references, citings, index terms

Multiscalar processors use a new, aggressive implementation paradigm for extracting large quantities of instruction level parallelism from ordinary high level language programs. A single program is divided into a collection of tasks by a combination of software and hardware. The tasks are distributed to a number of parallel processing units which reside within a processor complex. Each of these units fetches and executes instructions belonging to its assigned task. The appearance of a single log ...

5 Multiscalar processors

Gurindar S. Sohi, Scott E. Breach, T. N. Vijaykumar
August 1998 **25 years of the international symposia on Computer architecture (selected papers) ISCA '98**

Publisher: ACM Press

Full text available: pdf(1.57 MB) Additional Information: full citation, references, citings, index terms

6 Link-time binary rewriting techniques for program compaction

Bjorn De Sutter, Bruno De Bus, Koen De Bosschere
September 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 5

Publisher: ACM Press

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Small program size is an important requirement for embedded systems with limited amounts of memory. We describe how link-time compaction through binary rewriting can achieve code size reductions of up to 62&percent; for statically bound languages such as C, C++, and Fortran, without compromising on performance. We demonstrate how the limited amount of information about a program at link time can be exploited to overcome overhead resulting from separate compilation. This is done with sc ...

Keywords: Program representation, binary rewriting, code abstraction, compaction, interprocedural analysis, linker, whole-program optimization

7 Optimization: A sink-n-hoist framework for leakage power reduction

Yi-Ping You, Chung-Wen Huang, Jenq Kuen Lee

September 2005 **Proceedings of the 5th ACM international conference on Embedded software EMSOFT '05**

Publisher: ACM Press

Full text available: pdf(290.29 KB) Additional Information: full citation, abstract, references, index terms

Power leakage constitutes an increasing fraction of the total power consumption in modern semiconductor technologies. Recent research efforts have tried to integrate architecture and compiler solutions to employ power-gating mechanisms to reduce leakage power. This approach is to have compilers perform data-flow analysis and insert instructions at programs to shut down and wake up components whenever appropriate for power reductions. While this approach has been shown to be effective in early st ...

Keywords: balanced scheduling, compilers for low power, data-flow analysis, leakage power reduction, power-gating mechanisms

8 Architecture/power: Power efficient branch prediction through early identification of branch addresses

Chengmo Yang, Alex Orailoglu

October 2006 **Proceedings of the 2006 international conference on Compilers, architecture and synthesis for embedded systems CASES '06**

Publisher: ACM Press

Full text available: pdf(247.99 KB) Additional Information: full citation, abstract, references, index terms

Ever increasing performance requirements have elevated deeply pipelined architectures to a standard even in the embedded processor domain, requiring the incorporation of dynamic branch prediction subsystems to hide the execution latency of control-altering instructions. In this paper a low power early branch identification technique which enables the design of extremely power-efficient branch predictors and BTBs is proposed. Through static extraction of program information regarding the distance ...

Keywords: application-specific processors, dynamic branch prediction, low-power design

9 A fast, memory-efficient register allocation framework for embedded systems

Sathyanarayanan Thammanur, Santosh Pande

November 2004 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 26 Issue 6

Publisher: ACM Press

Full text available: pdf(1.01 MB) Additional Information: full citation, abstract, references, index terms

In this work, we describe a "just-in-time," usage density-based register allocator geared toward embedded systems with a limited general-purpose register set wherein speed, code size, and memory requirements are of equal concern. The main attraction of the allocator is that it does not make use of the traditional live range and interval analysis nor does it perform advanced optimizations based on range splitting but results in very good code quality. We circumven ...

Keywords: Code generation, compiler optimizations, compilers, dynamic compilation, embedded systems, register allocation

10 Incremental Commit Groups for Non-Atomic Trace Processing

Matt T. Yourst, Kanad Ghose

November 2005 **Proceedings of the 38th annual IEEE/ACM International Symposium on Microarchitecture MICRO 38**

Publisher: IEEE Computer Society

Full text available: pdf(614.37 KB) Publisher Site

Additional Information: full citation, abstract, index terms

We introduce techniques to support efficient non-atomic execution of very long traces on a new binary translation based, x86-64 compatible VLIW microprocessor. Incrementally committed long traces significantly reduce wasted computations on exception induced rollbacks by retaining the correctly committed parts of traces. We divide each scheduled trace into multiple commit groups; groups are committed to the architectural state after all instructions within and prior to each group complete without ...

Keywords: binary translation, VLIW, commitment,trace prediction

11 Sifting out the mud: low level C++ code reuse

Bjorn De Sutter, Bruno De Bus, Koen De Bosschere

November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '02**, Volume 37 Issue 11

Publisher: ACM Press

Full text available: pdf(1.35 MB)

Additional Information: full citation, abstract, references, citings, index terms

More and more computers are being incorporated in devices where the available amount of memory is limited. This contrasts with the increasing need for additional functionality and the need for rapid application development. While object-oriented programming languages, providing mechanisms such as inheritance and templates, allow fast development of complex applications, they have a detrimental effect on program size. This paper introduces new techniques to reuse the code of whole procedures at t ...

Keywords: code compaction, code size reduction

12 Affix grammar driven code generation

Mahadevan Ganapathi, Charles N. Fischer

October 1985 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 7 Issue 4

Publisher: ACM Press

Full text available: pdf(3.19 MB)

Additional Information: full citation, abstract, references, citings, index terms, review

Affix grammars are used to describe the instruction set of a target architecture for purposes of compiler code generation. A code generator is obtained automatically for a compiler using attributed parsing techniques. A compiler built on this model can automatically perform most popular machine-dependent optimizations, including peephole optimizations. Code generators based on this model demonstrate retargetability for the VAX1-11, iAPX2

13 Basic compiler algorithms for parallel programs

Jaejin Lee, David A. Padua, Samuel P. Midkiff

May 1999 **ACM SIGPLAN Notices , Proceedings of the seventh ACM SIGPLAN symposium on Principles and practice of parallel programming PPoPP '99**, Volume 34 Issue 8

Publisher: ACM Press

Full text available: pdf(1.45 MB) Additional Information: full citation, abstract, references, citings, index terms

Traditional compiler techniques developed for sequential programs do not guarantee the correctness (sequential consistency) of compiler transformations when applied to parallel programs. This is because traditional compilers for sequential programs do not account for the updates to a shared variable by different threads. We present a concurrent static single assignment (CSSA) form for parallel programs containing cobegin/coend and parallel do constructs and post/wait synchronization primitives. ...

14 Intraprogram dynamic voltage scaling: Bounding opportunities with analytic modeling

Fen Xie, Margaret Martonosi, Sharad Malik

September 2004 **ACM Transactions on Architecture and Code Optimization (TACO)**, Volume 1 Issue 3

Publisher: ACM Press

Full text available: pdf(980.11 KB) Additional Information: full citation, abstract, references, citings, index terms

Dynamic voltage scaling (DVS) has become an important dynamic power-management technique to save energy. DVS tunes the power-performance tradeoff to the needs of the application. The goal is to minimize energy consumption while meeting performance needs. Since CPU power consumption is strongly dependent on the supply voltage, DVS exploits the ability to control the power consumption by varying a processor's supply voltage and clock frequency. However, because of the energy and time overhead asso ...

Keywords: Analytical model, compiler, dynamic voltage scaling, low power, mixed-integer linear programming

15 Reducing code size through address register assignment

G. Chen, M. Kandemir, M. J. Irwin, J. Ramanujam

February 2006 **ACM Transactions on Embedded Computing Systems (TECS)**, Volume 5 Issue 1

Publisher: ACM Press

Full text available: pdf(942.62 KB) Additional Information: full citation, abstract, references, index terms

In DSP processors, minimizing the amount of address calculations is critical for reducing code size and improving performance, since studies of programs have shown that instructions that manipulate address registers constitute a significant portion of the overall instruction count (up to 55%). This work presents a compiler-based optimization strategy to "reduce the code size in embedded systems." Our strategy maximizes the use of indirect addressing modes with postincrement/de ...

Keywords: DSP, Software compilation, address registers, register assignment

16 Mapping esterel onto a multi-threaded embedded processor

Xin Li, Marian Boldt, Reinhard von Hanxleden

October 2006 **ACM SIGARCH Computer Architecture News , ACM SIGOPS Operating Systems Review , ACM SIGPLAN Notices , Proceedings of the 12th international conference on Architectural support for programming languages and operating systems ASPLOS-XII**, Volume 34 , 40 , 41 Issue 5 , 5 , 11

Publisher: ACM Press

Full text available: pdf(490.15 KB) Additional Information: full citation, abstract, references, index terms

The synchronous language Esterel is well-suited for programming control-dominated reactive systems at the system level. It provides non-traditional control structures, in

particular concurrency and various forms of preemption, which allow to concisely express reactive behavior. As these control structures cannot be mapped easily onto traditional, sequential processors, an alternative approach that has emerged recently makes use of special-purpose reactive processors. However, the designs propose ...

Keywords: concurrency, esterel, low-power processing, multi-threading, reactive systems

17 Operator strength reduction

Keith D. Cooper, L. Taylor Simpson, Christopher A. Vick
September 2001 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 23 Issue 5
Publisher: ACM Press

Full text available: pdf(240.36 KB) Additional Information: full citation, abstract, references, citings, index terms

Operator strength reduction is a technique that improves compiler-generated code by reformulating certain costly computations in terms of less expensive ones. A common case arises in array addressing expressions used in loops. The compiler can replace the sequence of multiplies generated by a direct translation of the address expression with an equivalent sequence of additions. When combined with linear function test replacement, strength reduction can speed up the execution of loops containing ...

Keywords: loops, static single assignment form, strength reduction

18 Automated analysis: Control-flow integrity

Martín Abadi, Mihai Budiu, Úlfar Erlingsson, Jay Ligatti
November 2005 **Proceedings of the 12th ACM conference on Computer and communications security CCS '05**
Publisher: ACM Press

Full text available: pdf(218.60 KB) Additional Information: full citation, abstract, references, citings, index terms

Current software attacks often build on exploits that subvert machine-code execution. The enforcement of a basic safety property, Control-Flow Integrity (CFI), can prevent such attacks from arbitrarily controlling program behavior. CFI enforcement is simple, and its guarantees can be established formally even with respect to powerful adversaries. Moreover, CFI enforcement is practical: it is compatible with existing software and can be done efficiently using software rewriting in commodity syste ...

Keywords: binary rewriting, control-flow graph, inlined reference monitors, vulnerabilities

19 Architectural support for software-based protection

Mihai Budiu, Úlfar Erlingsson, Martín Abadi
October 2006 **Proceedings of the 1st workshop on Architectural and system support for improving software dependability ASID '06**
Publisher: ACM Press

Full text available: pdf(642.62 KB) Additional Information: full citation, abstract, references, index terms

Control-Flow Integrity (CFI) is a property that guarantees program control flow cannot be subverted by a malicious adversary, even if the adversary has complete control of data memory. We have shown in prior work how CFI can be enforced by using inlined software guards that perform safety checks. The first part of this paper shows how modest Instruction Set Architecture (ISA) support can replace such guard code with single

instructions.On the foundation of CFI we have implemented XFI: a protecti ...

Keywords: binary rewriting, control-flow graph, control-flow integrity, hardware support, memory protection, security, software fault isolation

20 Parallelizing nonnumerical code with selective scheduling and software pipelining



Soo-Mook Moon, Kemal Ebcioğlu
November 1997 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 19 Issue 6
Publisher: ACM Press

Full text available: pdf(543.93 KB) Additional Information: full citation, abstract, references, citings, index terms

Instruction-level parallelism (ILP) in nonnumerical code is regarded as scarce and hard to exploit due to its irregularity. In this article, we introduce a new code-scheduling technique for irregular ILP called "selective scheduling" which can be used as a component for superscalar and VLIW compilers. Selective scheduling can compute a wide set of independent operations across all execution paths based on renaming and forward-substitution and can compute availab ...

Keywords: VLIW, global instruction scheduling, instruction-level parallelism, software pipelining, speculative code motion, superscalar

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2007 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player